WHAT MAKES FOR A HIT POP SONG?

NICK BORG AND GEORGE HOKKANEN

Overview

The possibility of a hit song prediction algorithm is both academically interesting and industry motivated. Several attempts have been made in the past with very little success (Pachet, [1], is a very good testament to this). The question we are interested in is to what extent features extracted from the music itself can be used to classify popular music, and in addressing it we also question the social factors that dictate popularity. One of these factors, discussed in [19] is preferential attachment, the idea that once an artist becomes popular enough it is more likely that the number of plays of their tracks becomes skewed in their favor despite different songs not all conforming to the type of music that made the artist popular. Depending on the success of this feature/popularity correlation, we may be motivated after the milestone to attempt correlations of popularity (via billboard charts and youtube views) with twitter data, but this is a consideration for the future.

Dataset

We are using the million song dataset subset which consists of ten thousand songs with metadata, chroma, and mfcc-like coefficients. The dataset is distributed by labrosa for free use [3]. One of the features given in the dataset is a popularity value called 'hotttnesss' produced by the Echonest, but this value appears too sparse in the data to be useful. For a popularity replacement measure, we have created a dataset of youtube view counts (coordinated with the subset only at the moment), ratings, and average rating of the first result for each query of artist name, song name. We checked by hand the accuracy of this scraping method and concluded that the two errors in thirty randomly drawn songs was not a problem given that the errors also coordinate well with very low view counts (i.e. something unpopular enough to not return a copy of the song on youtube ends up returning an unrelated video with a low view count).

Preliminary Analysis

The very first analyses we ran to get familiar with our data were basic correlation coefficients between different parts of the metadata and also with our extracted youtube view counts. The results were largely insignificant and included weak correlations such as one of .2 between the tempo and loudness metadata features. Correlations between the youtube view counts and the echonest metadata features loudness, tempo, hotttness, and danceability were completely negligible (less than .05 in magnitude). The fact that these are not at all correlated is interesting in its own right because it points to no single metadata feature being at all a good predictor of views on youtube.

First, we tried a shot in the dark technique of simply taking some amount of mfcc's (usually less than 100) and concatenating them into a vector of length less than 1200. Coordinating these with the youtube view counts obtained through scraping, we tried SVMs with several different kernels all to no avail (the accuracy was always less than .6). Our biggest hypothesis as to why this was a failed use of the mfcc data, which we think should very well describe a song, is that it entirely neglects the temporal progression of the sound. For this reason, we thought that the only intelligible way to use an SVM might be something like a string kernel with a symbolic representation of the sound. This is outlined in feature extraction and classification.

Feature Extraction

Given that the mfcc and spectral data is temporal, we wanted to use the ordering therein to describe the sound. Motivating a string kernel SVM approach, we creating 'string' features for our songs as follows.

For each i, we take the spectral bucket i (corresponding to a frequency-aggregate magnitude) for each spectra vector within a range of each song (usually about 45 seconds in the middle). This gives a list of values which correspond to the magnitudes over time of this slice of the sound spectrum for every song.

Next, using a subset of this data (we used two hundred of the ten thousand songs) we compute a list of intervals (we used 26 of them) that uniformly distribute the data. Then using these intervals, we compute a string for each sequence of data obtained in the first step by replacing each value with a symbol or letter that represents the interval.

What this method gives

Classification

SVM on xyz. Naive bayes on tags.

Results

Acknowledgements

[1] Pachet, F. and Roy, P. (2008) Hit Song Science is Not Yet a Science. Proceedings of Ismir 2008, pages 355-360, Philadelphia, USA

[2] Salganik, M. J. Dodds, P. S. Watts, D. J. Experimental Study of Inequality and Unpredictability in an Artificial Cultural Market, Science, 311, 854-856, 2006

[3] Million Song Dataset, official website by Thierry Bertin-Mahieux, available at: http://labrosa.ee.columbia.edu/millionsong/

[4] LIBSVM citation to go here